PROLOG: SUBSUMPTION OF EQUALITY AXIOMS BY THE HOMOGENEOUS FORM

E. W. ELCOCK

▷ The paper analyses the application of the homogeneous form of PROLOG programs to programs with equality. It is shown that the transformation to homogeneous form directly subsumes the transitivity and predicate substitutivity axioms of equality. Subsumption of the remaining axioms of equality within the same general framework is also considered.

1. PREAMBLE

For reasons that are well articulated in a number of sources [16, 17, 19, 18, 20] there is a considerable interest in systems which obviate the need for explicit axioms for equality by appropriately building them into the inference mechanism.

The paper shows how, for PROLOG [3] with equality, a simple syntactic transformation of the original PROLOG program allows certain axioms to be built in, while retaining SLD resolution as the inference mechanism.

There has also been interest in using syntactic transformations to provide a logical basis to establish the soundness of equational systems. In particular Hansson and Haridi [11] and, independently, van Emden and Lloyd [21] use the relevant part of the transformation discussed in this paper to establish the soundness of PROLOG-II [4] with respect to a certain equality theory. In using a program transformation for this purpose, it would seem important to recognize that certain equality axioms may be subsumed by the transformation itself, which therefore necessarily carries with it an implicit "kernel" equality theory. This is certainly the case for the so-called "homogeneous transformation" used in the analysis of PROLOG-II cited above.

Address correspondence to Professor E. W. Elcock, Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6A 5B7.

Received 5 March 1987; accepted 10 September 1987.

THE JOURNAL OF LOGIC PROGRAMMING

655 Avenue of the Americas, New York, NY 10010 0743-1066/89/$3.50

2. INTRODUCTION

Let $\underline{L}$ be a first order language with equality. Let the equality theory, $\underline{E}$, be

1. $x = x \leftarrow$ (reflexivity, $\underline{R}$)
2. $x = y \leftarrow y = x$ (symmetry, $\underline{S}$)
3. $x = z \leftarrow x = y, y = z$ (transivity, $\underline{T}$)
4. $p(x_1, \ldots, x_n) \leftarrow p(y_1, \ldots, y_n), x_1 = y_1, \ldots, x_n = y_n$ (predicate substitutivity, $\underline{PS}$)

where the axiom schema 4 applies to all predicates in $\underline{L}$ other than " = ".

Let $P \cup E$ be a program, where E is the set of program clauses having an equation as the head atom.

Let G be a goal.

Let H be the transformation defined as follows: If c is the program clause

$$p(t_1, \ldots, t_n) \leftarrow B,$$

then $H(c)$ is the clause

$$p(x_1, \ldots, x_n) \leftarrow x_1 = t_1, x_2 = t_2, \ldots, x_n = t_n, B,$$

where p may be equality (i.e., the head atom may be an equation), and B may be empty.

$H(P)$ is the set $\{H(c) : c \in P\}$.

$H(E)$ is the set $\{H(c) : c \in E\}$.

The result to be proved. Let $P \cup E$ be a program and G a goal. Then there exists an *SLD* refutation of

$$H(P) \cup H(E) \cup \{\underline{R}, \underline{S}\} \cup \{G\}$$

iff there exists an *SLD* refutation of

$$P \cup E \cup \underline{E} \cup \{G\}.$$

The transformation of P and E subsumes the equality theory $\underline{E} - \{\underline{R}, \underline{S}\}$, i.e., the axioms of transitivity and predicate substitutivity of $\underline{E}$.

The transformation H is an extension of van Emden and Lloyd's "homogeneous form" [21] to "recursive" equality theories. In [21], the first order language for the program and goal does not contain the predicate " = ". The equality theory is a *separable* theory. The result above applies quite generally to programs which allow equations anywhere in a clause, including the head of a clause. An auxiliary equality theory must be thought of as contributing to a "recursive" theory. This is the usual situation in mathematics. The subsumption result would of course be true for the "degenerate" case of separable theories ($E = \varnothing$).

STRUCTURE OF THE PROOF. We will prove a sequence of three lemmas, the first and third of which subsume a further axiom from the residual equality theory, with the second a "bridging" lemma: the residual equality theories after Lemmas 1 and 3 are $\underline{E} - \underline{PS}$, $\underline{E} - \underline{PS} - \underline{T} \equiv \{\underline{R}, \underline{S}\}$ respectively.

3. ILLUSTRATIVE EXAMPLE

Before presenting the formal proofs, a simple illustrative example might be helpful.

Given the program

1. $p(a) \leftarrow$
2. $b = a \leftarrow$
3. $b = c \leftarrow$

and the goal

4. $\leftarrow p(c)$

we have the refutation

5.	$\leftarrow p(x), c = x$	by $\underline{PS}$
6.	$\leftarrow c = a$	by 1
7.	$\leftarrow c = y, \underline{y = a}$	by $\underline{T}$
8.	$\leftarrow c = b$	by 2
9.	$\leftarrow b = c$	by $\underline{S}$
10.	$\leftarrow \square$	by 3

The refutation makes use of the predicate substitutivity, transitivity, and symmetry axioms of $\underline{E}$.

Consider now the transformed program

1. $p(x) \leftarrow x = a$
2. $x = y \leftarrow x = b, y = a$
3. $x = y \leftarrow x = b, y = c$

with the same goal

4. $\leftarrow p(c)$.

We now have the refutation

5.	$\leftarrow c = a$	by 1
6.	$\leftarrow \underline{c = b}, a = a$	by 2
7.	$\leftarrow \underline{b = c}, a = a$	by $\underline{S}$
8.	$\leftarrow b = b, c = c, a = a$	by 3
9.	$\leftarrow \square$	by $\underline{R}$

This refutation makes no use of the predicate replacability and transitivity axioms.

4. LEMMA 1

Let $P \cup E$ be a program and G a goal. Then there exists an SLD refutation of

$$T: \quad P \cup E \cup \underline{E} \cup \{G\}$$

iff there exists an SLD refutation of

$$T': \qquad H(P) \cup E \cup (\underline{E} - \underline{PS}) \cup \{G\}.$$

The "if" result is straightforward. We simply have to show that $T \models T'$, since it then follows that T' is unsatisfiable only if T is unsatisfiable. The result then follows from the completeness of SLD resolution [2].

To show that $T \models T'$ it is sufficient to show that $T \models H(P)$. Let c' be an arbitrary clause in $H(P)$. Then c' is $H(c)$ for some clause c in P. Let ps be the predicate substitutivity axiom for c. We show that $\{c, ps\} \models c'$. The forms of c, ps, and c are:

$$\begin{array}{ll} c: & p(t_1, \ldots, t_n) \leftarrow B. \\ ps: & p(x_1, \ldots, x_n) \leftarrow p(y_1, \ldots, y_n), x_1 = y_1, \ldots, x_n = y_n \\ c': & p(x_1, \ldots, x_n) \leftarrow x_1 = t_1, \ldots, x_n = t_n, B. \end{array}$$

We can resolve $p(y_1, \ldots, y_n)$ in ps with the head of c to obtain c'. The entailment $\{c, ps\} \models c'$ now follows from the resolution theorem [2].

It remains to show that if there is a refutation of T then there is a refutation of T'. It is worth remarking that this cannot be proved by the method used above: it is in fact easy to demonstrate that $T' \not\models T$. Consider the following set T':

$$\begin{array}{ll} c': & p(x) \leftarrow x = a, q(b) \\ \underline{R}: & x = x \leftarrow \\ \underline{S}: & x = y \leftarrow y = x \\ \underline{T}: & x = z \leftarrow x = y, y = z. \end{array}$$

The set T, in addition to containing $\underline{R}, \underline{S}, \underline{T}$, and the nonhomogeneous form c of c', contains the following predicate substitutivity axiom schema:

$$\underline{PS}: \qquad p(x) \leftarrow p(y), x = y.$$

Clearly, if $T' \not\models \underline{PS}$ then $T' \not\models T$. We now provide an interpretation I in which $\underline{PS}$ is false but which is a model of T'. The interpretation I is as follows:

$$\begin{array}{ll} \textit{Domain} & \{a, b\} \\ = & \{\langle a, a\rangle, \langle b, b\rangle, \langle a, b\rangle, \langle b, a\rangle\} \\ p & \{a\} \\ q & \varnothing \\ a & a \\ b & b \end{array}$$

Clearly, axioms $\underline{R}, \underline{S}$, and $\underline{T}$ are true in I. Axiom c' is also true, simply because $q(a)$ is false. However, the following instance of $\underline{PS}$:

$$p(b) \leftarrow p(a), b = a$$

is false in I, and therefore so is $\underline{PS}$.

We will prove the "only if" result using a proof theoretic argument. We will consider only unary predicates. Because of the uniform treatment of all arguments of p in the transformation of a clause $p(s_1, \ldots, s_n) \leftarrow B$ by H, the generalization of the proof to predicates of any arity is trivial.

Let R be an SLD refutation of $P \cup E \cup \underline{E} \cup \{G\}$. Consider the following two cases.

Case 1. R contains no input clauses from $\underline{PS}$ for p. Let the goal clause be

g: $\leftarrow p(t), G.$

The first atom at some time must be resolved against an input clause from P of the form

c: $p(s) \leftarrow B.$

There is no loss of generality[1] in considering this to be the next inference step, giving rise to the goal

g': $(B, G)\sigma$

where σ is the mgu of $p(s)$ and $p(t)$.

Consider now the following fragment of a refutation R' of $H(P) \cup E \cup (\underline{E} - \underline{PS}) \cup \{G\}$. $H(P)$ contains the clause $H(c)$, i.e.,

c': $p(x) \leftarrow x = s, B.$

Resolving the first atom of g against c', we get

$\leftarrow t = s, B, G.$

Resolving the first atom against $\underline{R}$, we get

$\leftarrow (B, G)\sigma,$

i.e., g' of R.

Case 2. As before, consider a goal clause

g: $\leftarrow p(t), G.$

Let us call a substitution instance of $p(x)$ introduced by an application of $\underline{PS}$ to $p(t)$ a $\underline{PS}$ child of $p(t)$. Let us recursively define $\underline{PS}$ descendants of $p(t)$ in the obvious way.

Let R contain just n $(n > 0)$ applications of $\underline{PS}$ to the goal atom $p(t)$ and $n - 1$ $\underline{PS}$ descendants. There is no loss of generality in supposing that these n applications of $\underline{PS}$ are the next n derivation steps. The resultant goal clause will be

$\leftarrow p(y_n), y_{n-1} = y_n, \ldots, t = y_1, G.$

As in case 1, $p(y_n)$ is now resolved against c, giving

g': $y_{n-1} = s, y_{n-2} = y_{n-1}, \ldots, t = y_1, B, G.$

Consider now the following fragment of an SLD refutation R' of $H(P) \cup E \cup \{G\} \cup \{\underline{E} - \underline{PS}\}$. As before, $H(P)$ contains the clause c'. Resolving the first atom of g against c', we get

$\leftarrow t = s, B, G.$

The first atom can be elaborated by $n - 1$ applications of the transitivity axiom to

[1] Cf. the "independence of computation rule" in [15].

give

$y_{n-1} = s, y_{n-2} = y_{n-1}, \ldots, t = y_1, B, G,$

i.e., g' of R.

5. LEMMA 2

Let $P \cup E$ be a program and G a goal. Then there exists an SLD refutation of

$$T': \quad H(P) \cup E \cup (\underline{E} - \underline{PS}) \cup \{G\}$$

iff there exists an SLD refutation of

$$T'': \quad H(P) \cup H(E) \cup (\underline{E} - \underline{PS}) \cup \{G\}.$$

The proof of the "only if" result is by showing that $T' \models T''$. This proof is similar to that in Lemma 1. To show that $T'' \models T'$ it is sufficient to show that $T'' \models E$. Let c be an arbitrary clause in E. Let c be

$t = s \leftarrow B.$

T'' contains the clause

$$H(c): \quad x = y \leftarrow x = t, y = s, B.$$

Resolving the two equations in the body of this clause with $\underline{R}$, we derive the clause c.

6. LEMMA 3

Let $P \cup E$ be a program and G a goal. Then there exists an SLD refutation of

$$H(P) \cup H(E) \cup (\underline{E} - \underline{PS}) \cup \{G\}$$

iff there exists an SLD refutation of

$$H(P) \cup H(E) \cup (\underline{E} - \underline{PS} - \underline{T}) \cup \{G\}.$$

The "if" result is trivial. The proof of the "only if" result has a similar structure to the general case proof in Lemma 1.

Let R be a refutation of $H(P) \cup E \cup (\underline{E} - \underline{PS}) \cup \{G\}$. Let a goal clause be

$$g: \quad \leftarrow s = t, G.$$

Let R contain an elaboration of $s = t$ using n ($n > 0$) applications of $\underline{T}$. Without loss of generality these n applications of $\underline{T}$ could be taken as the next n derivation steps. It is straightforward to show that, independently of which descendant equations are selected for resolution with $\underline{T}$, the resulting goal is always an alphabetic variant of

$$g': \quad \leftarrow s = x_1, x_1 = x_2, \ldots, x_n = t, G.$$

Ultimately, these $n+1$ equations must be resolved against $n+1$ input clauses

$$c_i: \qquad p_1 = q_i \leftarrow B_i,$$

$1 \leq i \leq n+1$ (where, as in the other lemmas, we will take the reflexivity axiom to be a degenerate form). Again, without loss of generality we can perform these resolution steps from left to right.

For all $n > 0$ there are at least two equations. Resolving these two equations generates the goal

$$g_R'': \qquad (B_1, B_2, x_2 = x_3, \ldots, x_n = t, G)\sigma_1\sigma_2,$$

where σ_1 is an mgu of $s = x_1$ and $p_1 = q_1$, and σ_2 is an mgu of $q_1 = x_2$ and $p_2 = q_2$.

Consider now a fragment of an SLD refutation R' of $H(P) \cup H(E) \cup (\underline{E} - \underline{PS} - \underline{T}) \cup \{G\}$. $H(E)$ contains the clause

$$H(c_i): \qquad x = y \leftarrow x = p_i, y = q_i, B_i$$

for all i, $1 \leq i \leq n+1$.

As before, let the goal clause be

$$g: \qquad \leftarrow s = t, G.$$

Resolving $s = t$ with

$$c_1': \qquad x = y \leftarrow x = p_1, y = q_1, B_1$$

we get

$$\leftarrow s = p_1, t = q_1, B_1, G$$

Resolving $t = q_1$ with $\underline{S}$ we get

$$\leftarrow s = p_1, q_1 = t, B_1, G.$$

Resolving $q_1 = t$ with

$$c_2': \qquad x'' = y'' \leftarrow x'' = p_2, y'' = q_2, B_2,$$

we get

$$g_R'': \qquad s = p_1, q_1 = p_2, t = q_2, B_1, B_2, G.$$

If $n = 1$, then g_R'' is simply

$$\leftarrow (B_1, B_2, G)\sigma_1\sigma_2'$$

where σ_1 is an mgu of $s = x_1$ and $p_1 = q_1$, and σ'_2 is an mgu of $q_1 = t$ and $p_2 = q_2$. This goal is immediately derivable from g_R'' by resolving the three equations with $\underline{R}$.

If $n > 1$, then we simply return to the elaboration of the two derivations. We have

$$g_R'': \qquad \leftarrow (B_1, B_2, q_2 = x_3, x_3 = x_4, \ldots, x_n = t, G)\sigma_1\sigma_2.$$

Resolving $q_2 = x_3$ with c_3 we get

$$g_R''': \qquad (B_1, B_2, B_3, x_3 = x_4, \ldots, x_n = t, G)\sigma_1\sigma_2\sigma_3,$$

where σ_3 is an mgu of $q_2 = x_3$ and $p_3 = q_3$.

We continue the elaboration of g''_R by resolving $t = q_2$ against $\underline{S}$ and then resolving $q_2 = t$ against

$$H(c_3): \qquad x = y \leftarrow x = p_3, y = q_3, B_3.$$

We get

$$g'''_R \qquad s = p_1, q_1 = p_2, q_2 = p_3, t = q_3, B_1, B_2, B_3, G.$$

If $n = 2$ we can derive g'''_R from g''_R as before. If $n > 2$ we simply proceed with a further iteration in the development of the two derivations.

7. THE SUBSUMPTION THEOREM

Theorem. There exists an SLD refutation of

$$P \cup E \cup \underline{E}\{G\}$$

iff there exists an SLD refutation of

$$H(P) \cup H(E) \cup \{\underline{R}, \underline{S}\} \cup \{G\}.$$

We have established:

Lemma 1. There exists an SLD refutation of

$$P \cup E \cup \underline{E} \cup \{G\}$$

iff there exists an SLD refutation of

$$H(P) \cup E \cup (\underline{E} - \underline{PS}) \cup \{G\}.$$

Lemma 2. There exists an SLD refutation of

$$H(P) \cup E \cup (\underline{E} - \underline{PS}) \cup \{G\}$$

iff there exists a refutation of

$$H(P) \cup H(E) \cup (\underline{E} - \underline{PS}) \cup \{G\}.$$

Lemma 3. There exists an SLD refutation of

$$H(P) \cup H(E) \cup (\underline{E} - \underline{PS}) \cup \{G\}$$

iff there exists an SLD refutation of

$$H(P) \cup H(E) \cup (\underline{E} - \underline{PS} - \underline{T}) \cup \{G\}.$$

The theorem follows immediately from the three lemmas.

8. SUBSUMPTION OF OTHER AXIOMS OF EQUALITY

The symmetry axiom turns out to be easily subsumed by an interesting modification of the transformation H of Section 2. As before, we consider a program $P \cup E$. The transformation for clauses in P is unchanged. However, for clauses in E, if c is the clause

$$s = t \leftarrow B,$$

then $H(c)$ is the clause

$x = y \leftarrow s = x, y = t, B.$

The idea for this asymmetric treatment of the arguments of the head atom of a clause in E is due to Chan [1]. With this change it is not difficult to show that the transformation H now also subsumes the symmetry axiom. In [1] Chan gives a fixed point proof. It is not difficult to use the proof method of this paper to establish the same result.

Unfortunately, the way the subsumption takes place has unpleasant properties for SLD resolution, and for the details of the otherwise straightforward proof theoretic proof. Essentially an SLD refutation replicates bodies of clauses in E whose heads need to be symmetrized. Both this point, and the actual subsumption by the asymmetric treatment of the arguments of the head atom, are sufficiently intriguing that an illustrative example is given below.

Consider the program

1. $p(c) \leftarrow$
2. $a = b \leftarrow p(c)$

and goal

3. $\leftarrow a = b.$

A refutation is

4. $\leftarrow p(c)$ by 2
5. $\square$ by 1

Consider now the transformed program

1. $p(x) \leftarrow x = c$
2. $x = y \leftarrow a = x, y = b, p(c)$

with the same goal

3. $\leftarrow a = b.$

A refutation is

4. $\leftarrow a = a, b = b, p(c)$ by 2
5. $\leftarrow p(c)$ by $\underline{R}$
6. $\square$ by 1

Consider now a refutation of the program with the symmetrized goal

1. $p(c) \leftarrow$
2. $a = b \leftarrow p(c)$
3. $\leftarrow b = a$
4. $\leftarrow a = b$ by $\underline{S}$
5. $\leftarrow p(c)$ by 2
6. $\square$ by 1

A refutation of the transformed program with the same symmetrized goal is

1. $p(x) \leftarrow x = c$
2. $x = y \leftarrow a = x, y = b, p(c)$
3. $\leftarrow b = a$
4. $\leftarrow a = b, a = b, p(c)$ by 2
5. $\leftarrow a = a, b = b, p(c), a = a, b = b, p(c), p(c)$ by 2

...

□

In general, the three occurrences of the body of the clause in E would be alphabetic variants of each other. The presence of these variants is the source of an irritating tediousness in a proof of the lemma above. In any case, the multiple occurrences make the symmetry subsumption method intellectually displeasing, and in this paper we have preferred to present the more limited result.

There are other ways of treating the symmetry axiom. One method is to explicitly symmetrize E. Let

$$E_S: \qquad E \cup \{s = t, \leftarrow B | t = s \leftarrow B \in E\}.$$

If we now replace $H(E)$ in our result by $H(E_S)$, then, again, an explicit symmetry axiom is unnecessary, this time with the only cost that the number of program equality clauses is doubled.

Recently Demopoulos [6] has given a model theoretic proof of the subsumption result [13] for symmetrized programs. The model theoretic argument consists in constructing two (interacting) sequences of Herbrand models with the property that the limit of one sequence is a minimal model of both $P \cup E \cup \underline{E} \cup \{G\}$ and $H(P) \cup H(E_S) \cup \{\underline{R}, G\}$. This minimal model property establishes our subsumption result.

As proofs, the three proofs of the subsumption result would seem to be comparable in their degree of difficulty. A preference possibly would simply reflect technical familiarity and ease with a particular proof style. The proof theoretic proof does however offer insights into the pragmatics of an implementation.

The work by Cox and Pietrzykowski on surface deduction [5] defines a transformation on a set of Horn clauses, called *flattening*, which intuitively is a covert continuation of homogeneous form. Flattening subsumes transitivity, predicate substitutivity, and function substitutivity. Though their work is cast in a system consisting of two inference rules in addition to linear, binary input resolution, it is implicit in their results (see, for example, [7]) that the two additional inference rules can be omitted in favor of the reflexive axiom. A difficulty with their transformation is that the set of Horn clauses obtained from the transformation is not obviously amenable to use with PROLOG technology.

As far as functions are concerned, flattening is closely related to simply "unnesting" function composition by the use of auxiliary variables. This notion was used by Haridi [12] to simplify his proof procedure. It has been used by Elcock [8] to create a transformation that subsumes *all* the axioms and axiom schemas of standard equality. Unfortunately, like the work of Cox and Pietrzykowski, it is not easy to use the result in implementing sequential systems.

9. SUMMARY AND CONCLUSIONS

We have shown how to transform a PROLOG program P in a language with equality, to another PROLOG program P' in such a way that the equality relation in P' is transitive with the predicate substitutivity property. So transitivity and predicate substitutivity of equality are subsumed without modifying the standard inference machine of PROLOG. It is our belief that the transformation offers possibilities for realizing some of the search economies that pragmatically motivate building equational theories into the inference system [17].

Indeed, Kornfeld's pragmatic work [14] on equality for PROLOG has particularly close ties with a pragmatic system one might consider based on the work reported here. Kornfeld's extended unification mechanism can be interpreted nondeterministically. Informally the nondeterministic unification of a term t with a term s involves the selection of one of the operations: unifying t and s, generating the goal $t = s$, or generating the goal $s = t$. It is not difficult [9] to show that this nondeterministic generalization subsumes the axioms of symmetry, transitivity, and predicate substitutivity. Its relationship with the transformational approach of this paper is discussed in [10].

We have commented briefly on other proof methods and on the subsumption of additional axioms.

This current work is an extension and simplification of work reported in Hoddinott and Elcock [13], and the author would like to acknowledge the debt.

The author would like to thank his colleagues Mr. P. Hoddinott for the first collaborative elaboration of the result, and his colleagues Dr. E. P. Stabler, Jr., and Dr. K. H. Chan for constructive criticism on this recasting of the original proof to try to make it simpler and closer to one's intuitions about derivations. In addition, acknowledgements are in order to these colleagues and to Dr. W. Demopoulos for many helpful discussions to do with building in equational theories.

The work has been supported in part under NSERC operating grant number A2445, and in part by the IBM Corporation.

REFERENCES

1. Chan, K. H., Equivalent Logic Programs and Symmetric Homogeneous Forms of Logic Programs with Equality, Technical Report 86, Dept. of Computer Science, Univ. of Western Ontario, London, Ont., Canada, 1986; *Comput. Intell.*, to appear.
2. Clark, Keith L., Predicate Logic as a Computational Formalism, Technical Report, Dept. of Computing, Imperial College, London, 1979.
3. Clocksin, W. F. and Mellish, C. S., *Programming in* PROLOG, Springer, New York, 1981.
4. Colmerauer, A. et al., PROLOG II: Reference Manual and Theoretical Model, Technical Report, Groupe d'Intelligence Artificielle, Faculté des Sciences de Lumy, Marseilles, 1982.
5. Cox, P. T. and Pietrzykowski, T., Incorporating Equality into Logic Programming via Surface Deduction, *Ann. Pure Appl. Logic* 31: 177–189 (1986).
6. Demopoulos, W., The Homogeneous Form of Logic Programs with Equality, Private communication, 1986.

7. Elcock, E. W. and Hoddinott, P., A Comment on the Relation between *S*-Deduction and Linear Binary Input Resolution Together with the Reflexive Axiom, Technical Report TR-145, Dept. of Computer Science, Univ. of Western Ontario, 1985.
8. Elcock, E. W., A Simple Extension of Horn Clauses for Logic Programming with Equality, Technical Report TR 183, Dept. of Computer Science, Univ. of Western Ontario, 1986.
9. Elcock, E. W. and Hoddinott, P., Kornfeld's Extended Unification and Classical Equality, Technical Report TR 186, Dept. of Computer Science, Univ. of Western Ontario, 1986.
10. Elcock, E. W. and P. Hoddinott, Classical Equality and PROLOG, in: *Proceedings of the Sixth Canadian Conference on Artificial Intelligence, AI86*, 1986, pp. 103–108.
11. Hansson, A. and Haridi, S., Logic Programming in a Natural Deduction Framework, in: *Workshop on Functional Language and Computer Architecture,* 1981.
12. Haridi, A. S., Logic Programming Based on a Natural Deduction System, Ph.D. Thesis, Royal Institute of Technology, Stockholm, 1981.
13. Hoddinott, P. and Elcock, E. W., PROLOG: Subsumption of Equality Axioms by the Homogeneous Form, in: *Proceedings of the IEEE Symposium on Logic Programming*, 1986, pp. 115–126.
14. Kornfeld, W. A., Equality for PROLOG, in: *Proceedings, Seventh International Joint Conference on Artificial Intelligence,* 1983 pp. 514–519.
15. Lloyd, J. W., *Foundations of Logic Programming,* Springer, New York, 1984.
16. Loveland, D. W., *Automated Theorem Proving: A Logical Basis,* North-Holland, New York, 1978.
17. Plotkin, G., Building-in Equational Theories, in: *Machine Intelligence 7,* Wiley, 1972, pp. 73–90.
18. Robinson, J. A., The Generalized Resolution Principle, in: *Machine Intelligence 3,* American Elsevier, New York, 1968, pp. 77–93.
19. Robinson, G. and Wos, L. T., in: Paramodulation and First Order Theorem Proving with Equality, in: *Machine Intelligence 4,* Edinburgh U.P., 1969, pp. 135–150.
20. Sibert, E. E., A Machine-Oriented Logic Incorporating the Equality Relation, in: *Machine Intelligence 4,* Edinburgh U.P., 1969 pp. 103–133.
21. van Emden, M. H. and Lloyd, J. W., A Logical Reconstruction of PROLOG II, *J. Logic Programming* 1(2):143–150 (Aug. 1984).